

Отчёта по лабораторной работе №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Барето Вилиан Мануел

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файлы листинга	11
3.3	Задание для самостоятельной работы	13
4	Выводы	17

Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
3.2	Заполняем файл	7
3.3	Запускаем файл и смотрим на его работу	7
3.4	Изменяем файл	8
3.5	Запускаем файл и смотрим на его работу	8
3.6	Редактируем файл	9
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	9
3.8	Создаем файл командой <code>touch</code>	9
3.9	Заполняем файл	10
3.10	Смотрим на работу программ	10
3.11	Создаем файл листинга	11
3.12	Изучаем файл	11
3.13	Удаляем операндум из файла	12
3.14	Транслируем файл	12
3.15	Изучаем файл с ошибкой	13
3.16	Создаем файл командой <code>touch</code>	13
3.17	Пишем программу	14
3.18	Смотрим на работу программы(всё верно)	14
3.19	Создаем файл командой <code>touch</code>	15
3.20	Пишем программу	15
3.21	Проверяем работу программы	15
3.22	Проверяем работу программы	16

1 Цель работы

Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

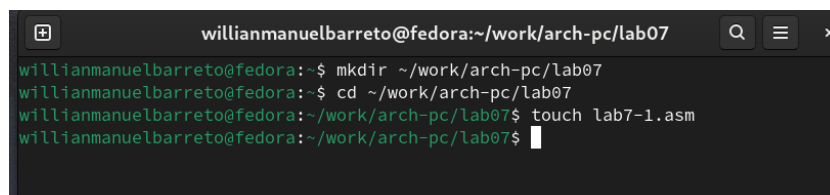
2 Задание

Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

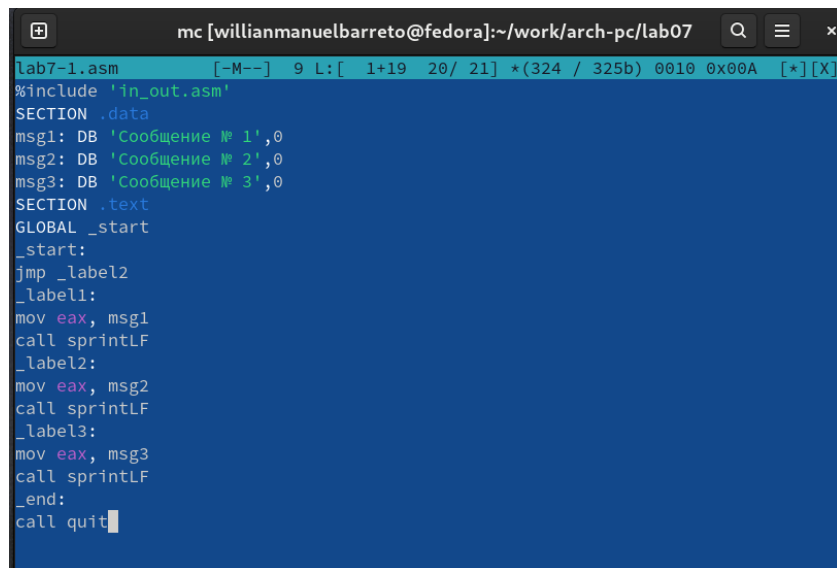
Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. fig. 3.1).



```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07
willianmanuelbarreto@fedora:~$ mkdir ~/work/arch-pc/lab07
willianmanuelbarreto@fedora:~$ cd ~/work/arch-pc/lab07
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

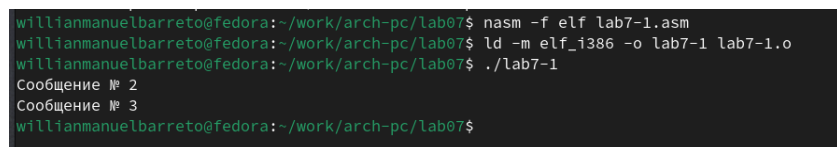
Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. fig. 3.2).

A screenshot of a text editor window titled 'mc [willianmanuelbarreto@fedora]:~/work/arch-pc/lab07'. The editor shows the content of 'lab7-1.asm'. The code includes a header, data section with three messages in Russian, and a text section with assembly instructions to print these messages and then quit.

```
lab7-1.asm [-M--] 9 L:[ 1+19 20/ 21] *(324 / 325b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1
call sprintf
_label2:
mov eax, msg2
call sprintf
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

Рис. 3.2: Заполняем файл

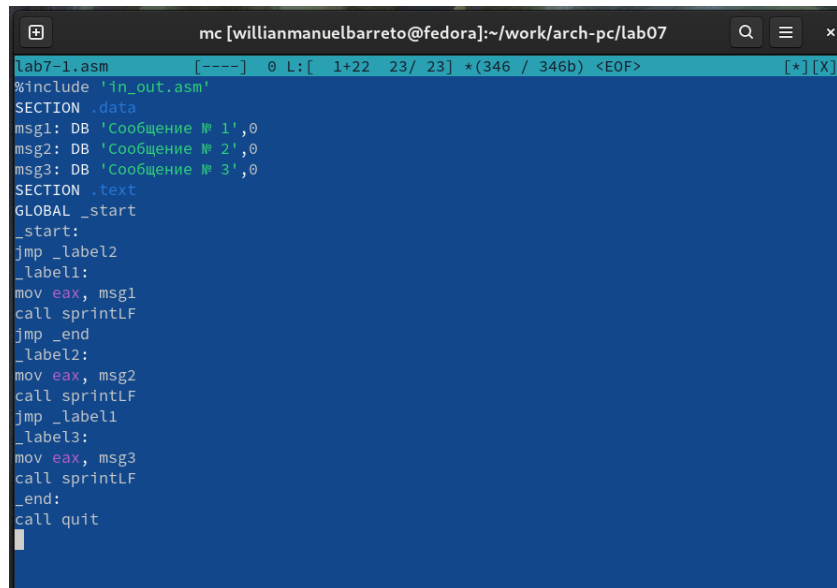
Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

A screenshot of a terminal window showing the compilation and execution of the assembly file. The user runs 'nasm -f elf lab7-1.asm', then 'ld -m elf_i386 -o lab7-1 lab7-1.o', and finally './lab7-1'. The output shows the three messages from the assembly code.

```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.3: Запускаем файл и смотрим на его работу

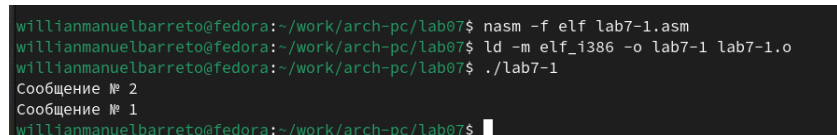
Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. fig. 3.4).



```
lab7-1.asm [----] 0 L: [ 1+22 23/ 23] *(346 / 346b) <EOF> [*] [X]
#include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1
call sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

Рис. 3.4: Изменяем файл

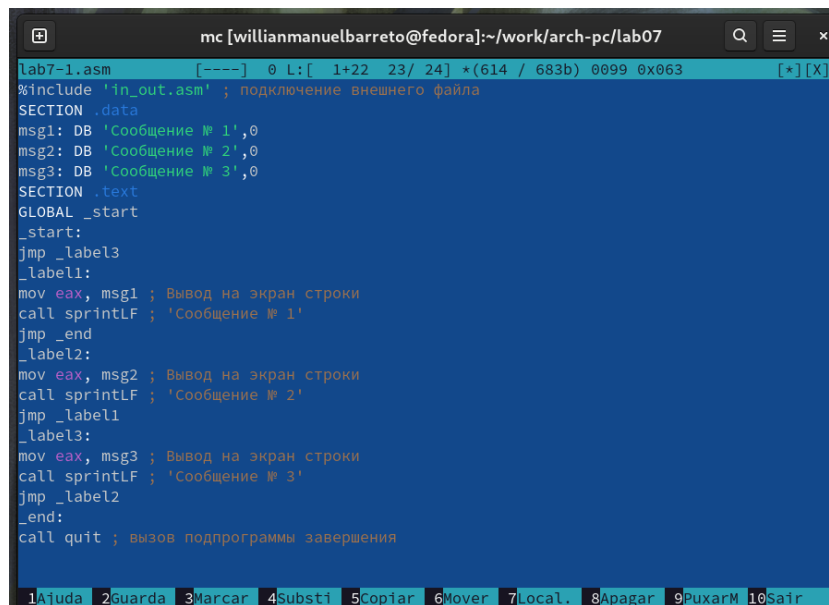
Создаем исполняемый файл и запускаем его (рис. fig. 3.5).



```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. fig. 3.6).



```
lab7-1.asm [----] 0 L: [ 1+22 23/ 24] *(614 / 683b) 0099 0x063 [*][X]
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.6: Редактируем файл

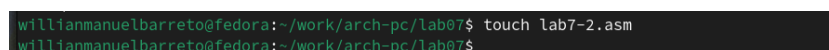
Создаем исполняемый файл и запускаем его (рис. fig. 3.7).



```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл (рис. fig. 3.8).



```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. fig. 3.9).

```

mc [willianmanuelbarreto@fedora]:~/work/arch-pc/lab07
lab7-2.asm  [----] 55 L: [ 1+28 29/ 49] *(925 /1657b) 0010 0x00A [*][X]
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
1Ajuda 2Guarda 3Marcar 4Substitui 5Copiar 6Mover 7Localizar 8Apagar 9Puxar Menu 10Sair

```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения B (рис. fig. 3.10).

```

willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$

```

Рис. 3.10: Смотрим на работу программ

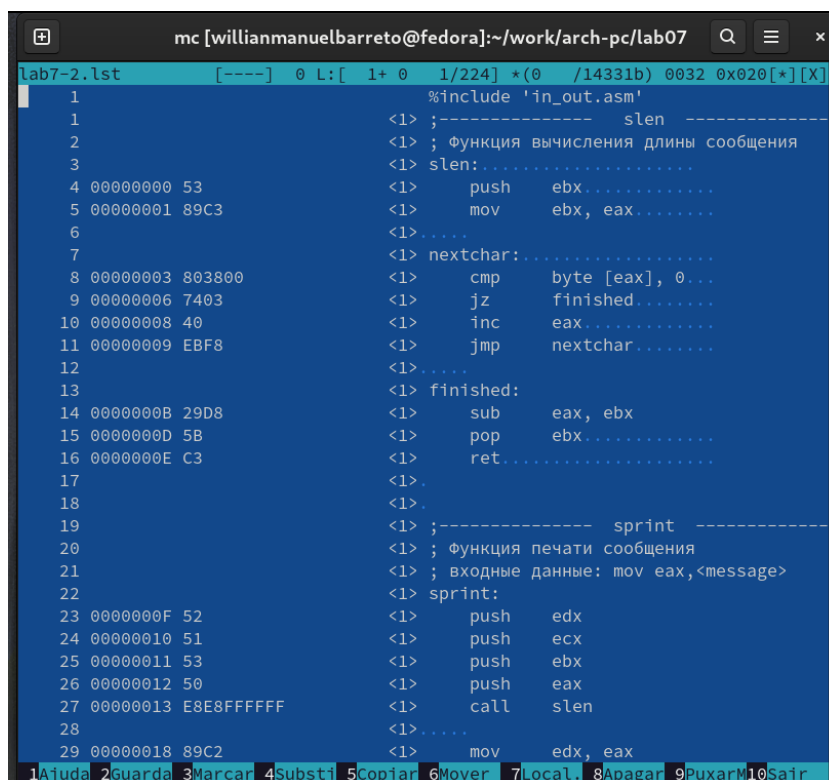
3.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. fig. 3.11).

```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.11: Создаем файл листинга

Открываем файл листинга с помощью команды `mcedit` и изучаем его (рис. fig. 3.12).



```
lab7-2.lst  [----]  0 L:[ 1+ 0 1/224] *(0 /14331b) 0032 0x020[*][X]
1                                     <1> ;----- slen -----
1                                     <1> ; Функция вычисления длины сообщения
2                                     <1> slen:.....
3                                     <1> push    ebx.....
4 00000000 53                         <1> mov     ebx, eax.....
5 00000001 89C3                       <1> .....
6                                     <1> nextchar:.....
7                                     <1> cmp     byte [eax], 0...
8 00000003 803800                     <1> jz      finished.....
9 00000006 7403                       <1> inc     eax.....
10 00000008 40                        <1> jmp     nextchar.....
11 00000009 EBF8                      <1> .....
12                                     <1> finished:
13                                     <1> sub     eax, ebx
14 0000000B 29D8                      <1> pop     ebx.....
15 0000000D 5B                        <1> ret.....
16 0000000E C3                       <1> .....
17                                     <1> .....
18                                     <1> ;----- sprint -----
19                                     <1> ; Функция печати сообщения
20                                     <1> ; входные данные: mov eax,<message>
21                                     <1> sprint:
22                                     <1> push    edx
23 0000000F 52                         <1> push    ecx
24 00000010 51                         <1> push    ebx
25 00000011 53                         <1> push    eax
26 00000012 50                         <1> call    slen
27 00000013 E8E8FFFFFF                <1> .....
28                                     <1> mov     edx, eax
29 00000018 89C2
```

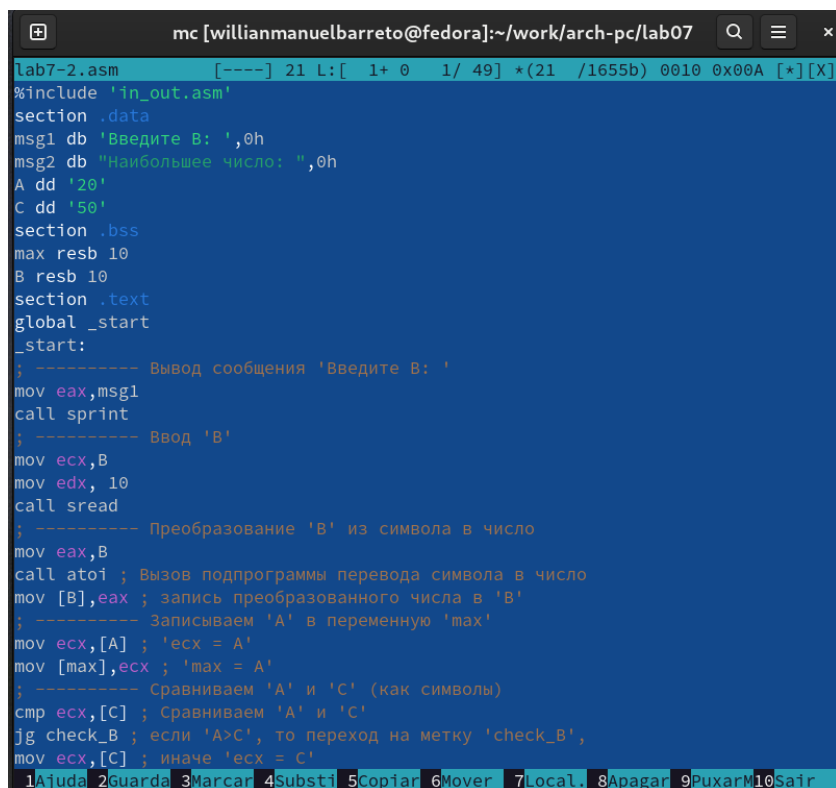
Рис. 3.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, B01000000-машинный код, `mov ebx,1`-присвоение переменной `ebx` значения 1.

Строка 34: 00000022-адрес в сегменте кода, B80400000-машинный код, `mov eax,4`-присвоение переменной `eax` значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

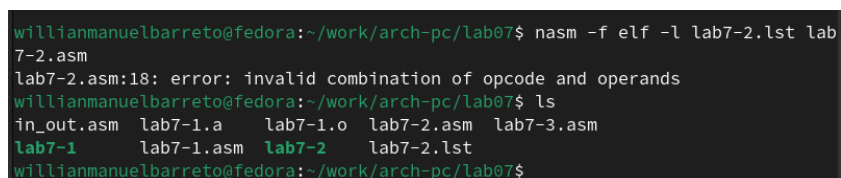
Открываем файл и удаляем один операндум (рис. fig. 3.13).



```
lab7-2.asm [----] 21 L:[ 1+ 0 1/ 49] *(21 /1655b) 0010 0x00A [*][X]
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx, 10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
1Ajuda 2Guarda 3Marcar 4Substi 5Copiar 6Mover 7Local. 8Apagar 9PuxarM10Sair
```

Рис. 3.13: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. fig. 3.14).



```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab
7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.a lab7-1.o lab7-2.asm lab7-3.asm
lab7-1 lab7-1.asm lab7-2 lab7-2.lst
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.14: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. fig. 3.15).

```

mc [willianmanuelbarreto@fedora]:~/work/arch-pc/lab07
lab7-2.lst [----] 40 L:[170+29 199/225] *(12299/14419b) 0109 0x06[*][X]
169 000000E5 CD80 <1> int 80h
170 000000E7 C3 <1> ret
2 section .data
3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00.....
4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000....
5 00000035 32300000 A dd '20'
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res Ah> max resb 10
9 0000000A <res Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 mov edx
18 ***** error: invalid combination of opcode a
19 000000F7 E847FFFFFF call sread
20 ; ----- Преобразование 'B' из симв
21 000000FC B8[0A000000] mov eax,B
22 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода
23 00000106 A3[0A000000] mov [B],eax ; запись преобразованного ч
1Ajuda 2Guarda 3Marcar 4Substitui 5Copiar 6Mover 7Localizar 8Apagar 9Puxar para 10Sair

```

Рис. 3.15: Изучаем файл с ошибкой

3.3 Задание для самостоятельной работы

ВАРИАНТ-20

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных \boxed{x} , \boxed{y} и \boxed{z} . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Создаем новый файл (рис. fig. 3.16).

```

willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$

```

Рис. 3.16: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе, 3е вводится из консоли) (рис. fig. 3.17).

```

lab7-3.asm  [----] 29 L: [ 1+29 30/ 50] *(960 /1743b) 0010 0x00A [*][X]
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '24'
C dd '15'
section .bss
min resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jnl check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
1Ajuda 2Guarda 3Marcar 4Substi 5Copiar 6Mover 7Local. 8Apagar 9PuxarM10Sair

```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. fig. 3.18).

```

willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 98
Наименьшее число: 24
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$

```

Рис. 3.18: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений x и y вычисляет значение заданной функции $f(x, y)$ и выводит результат вычислений. Вид функции $f(x, y)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений

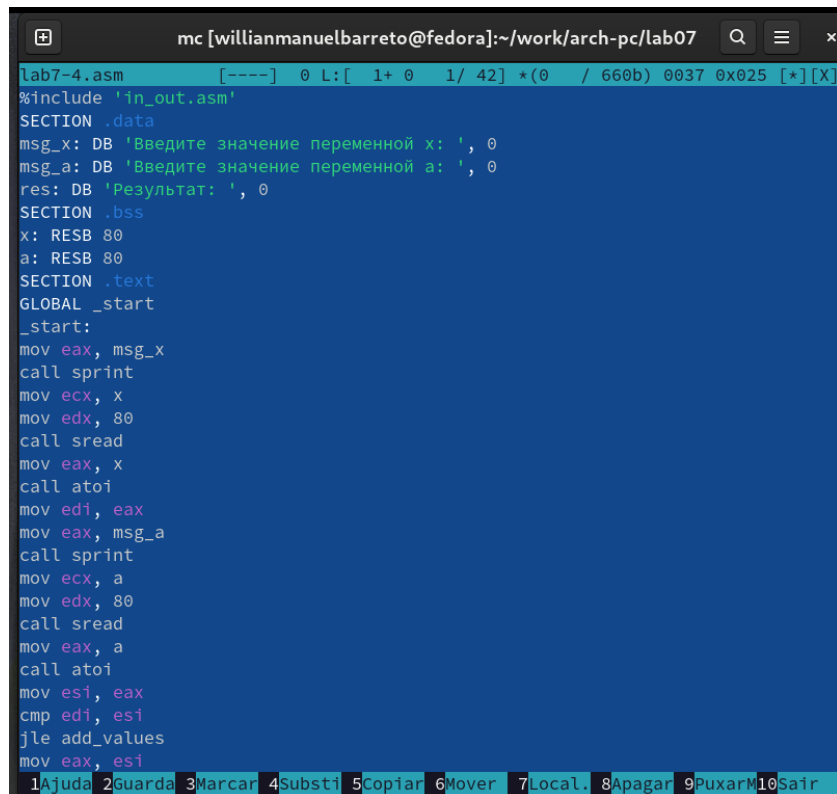
❌ и ❌ из 7.6.

Создаем новый файл (рис. fig. 3.19).

```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.19: Создаем файл командой touch

Открываем его и пишем программу, которая решит систему уравнений, при данных, введенных в консоль (рис. fig. 3.20).



```
mc [willianmanuelbarreto@fedora]:~/work/arch-pc/lab07
lab7-4.asm [----] 0 L: [ 1+ 0 1/ 42] *(0 / 660b) 0037 0x025 [*][X]
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax
cmp edi, esi
jle add_values
mov eax, esi
1Ajuda 2Guarda 3Marcar 4Substi 5Copiar 6Mover 7Local. 8Apagar 9PuxarM10Sair
```

Рис. 3.20: Пишем программу

Транслируем файл и проверяем его работу при x=1 и a=2(рис. fig. 3.21).

```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
```

Рис. 3.21: Проверяем работу программы

Транслируем файл и проверяем его работу при $x=2$ и $a=1$ (рис. fig. 3.22).

```
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
willianmanuelbarreto@fedora:~/work/arch-pc/lab07$
```

Рис. 3.22: Проверяем работу программы

4 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.